

RacerGS: Residual-Autoregressive Context for 3D Gaussian Splatting Compression

Wenqing Wang Huimin Zeng Yun Fu

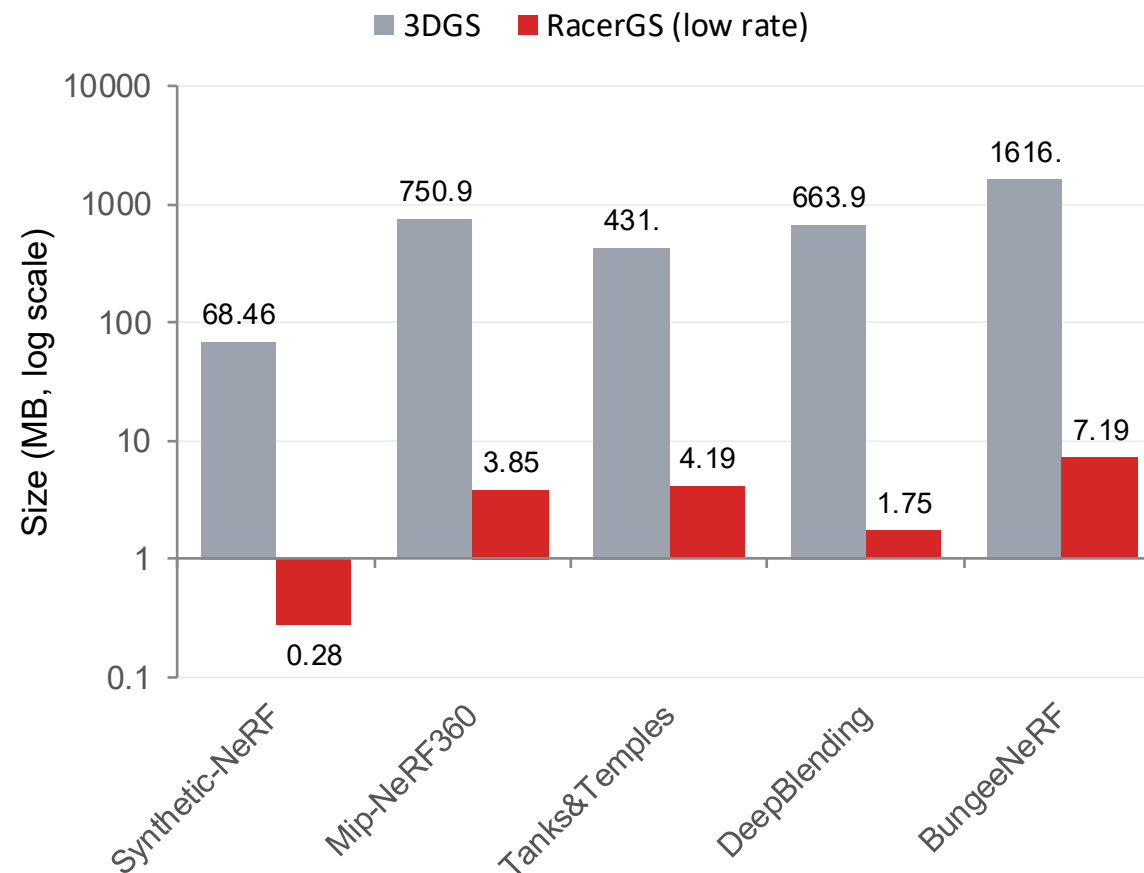
Northeastern University



Project Page

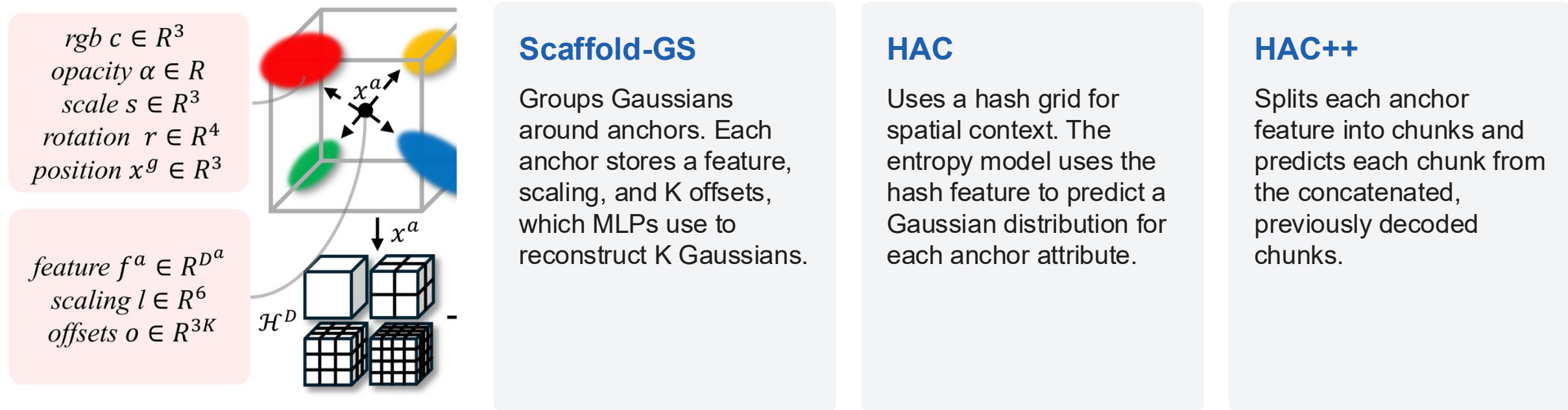
3DGS Renders Fast, but Scenes Are Large

- 3D Gaussian Splatting represents a scene with millions of Gaussians, each with a position, scale, rotation, opacity, and color.
- A single Mip-NeRF360 scene takes **~750 MB**, while a BungeeNeRF city scene takes **1.6 GB**.
- These sizes make storage, streaming, and mobile use difficult.
- RacerGS compresses these scenes to **a few MB** while preserving comparable rendering quality.



Over 150× smaller than 3DGS, and 28× smaller than Scaffold-GS, at comparable quality.

Background: Anchor-Based 3DGS Compression



- **Entropy-coded anchor attributes** take up most of the storage, so their bit cost largely determines file size.
- **Better context** leads to sharper predicted distributions and fewer bits per anchor.

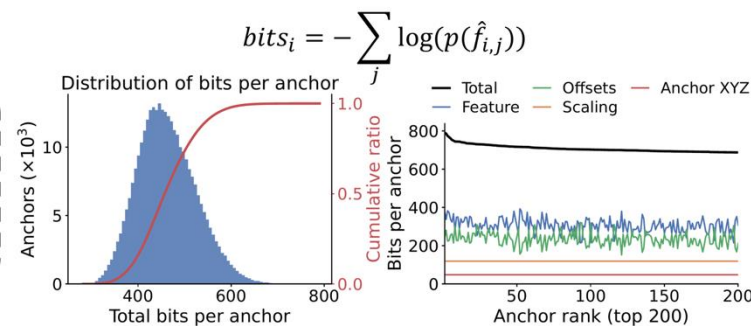
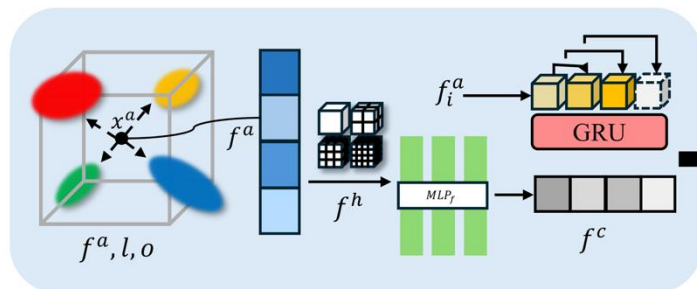
Where Do the Bits Go?

3D Gaussian Splatting

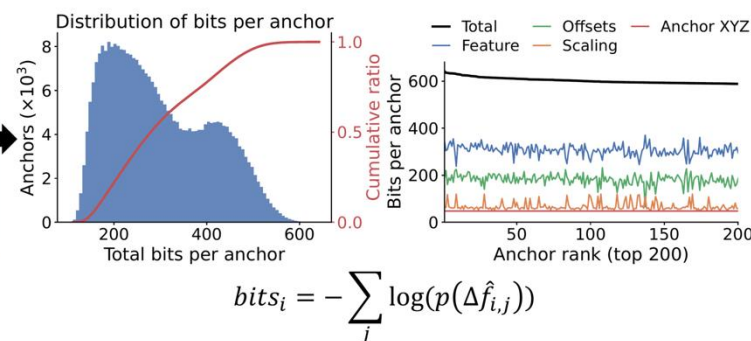


Previous Methods

- (1) Limited spatial context modeling
- (2) Inefficient entropy modeling
- (3) Lack of causal modeling across anchor feature groups



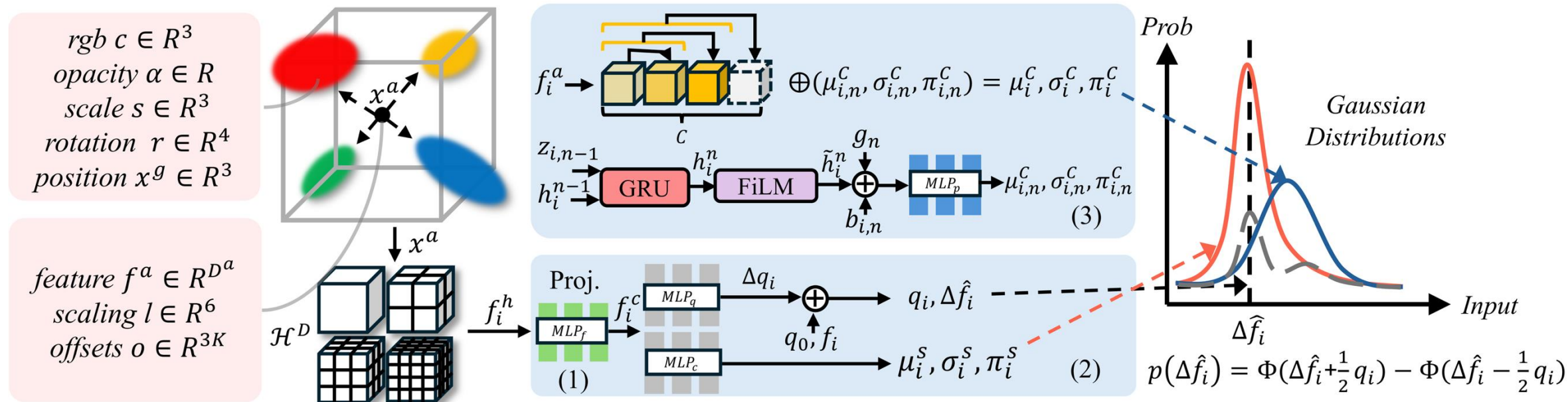
RacerGS



- **Limited spatial context.** Raw, high dimensional hash features increase the entropy model's parameter count and must serve both rendering and compression.
- **Inefficient entropy modeling.** Attributes are coded directly, without exploiting their small deviations from a context predicted mean.
- **No causal chunk model.** Concatenating decoded chunks keeps growing the input instead of summarizing the history.

RacerGS shifts the bits-per-anchor distribution left on the same scene.

RacerGS: Overview



(1) Hash projection model

Projects hash encodings into a compact context feature for the entropy model.

(2) Residual quantization

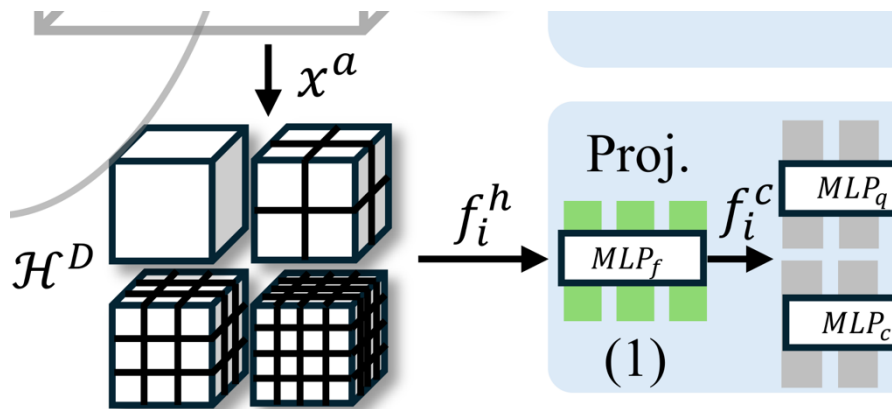
Encodes attribute residuals relative to context predicted means, using learned quantization steps.

(3) FiLM-modulated AR GRU

Refines each chunk's distribution using a fixed size summary of previously decoded chunks.

Predict each anchor better before coding it.

(1) Hash Projection Model



$$f_i^c = \text{MLP}_f(f_i^h) \in \mathbb{R}^{D^c}, \quad f_i^h = \text{Interp}(x_i^a, \mathcal{H}) \in \mathbb{R}^{D^h}$$

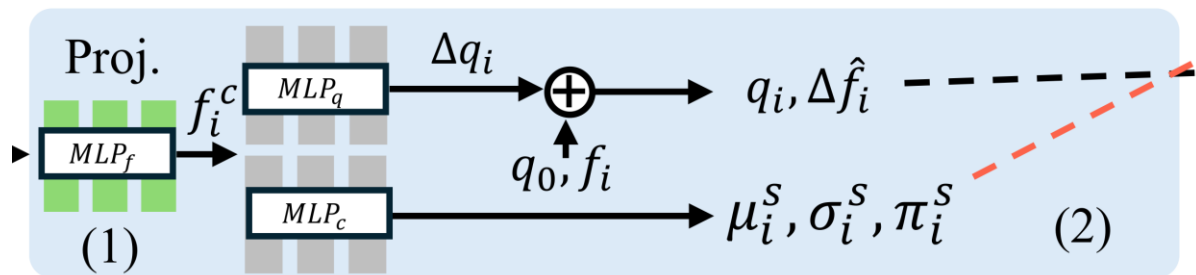
- Raw hash features make every entropy MLP's input width depend on the hash dimension, while one encoding serves both rendering and entropy modeling.
- A nonlinear MLP combines information across dimensions. Joint training with the entropy loss retains features that help reduce bit cost.
- The hash grid can grow without increasing the entropy model's parameter count.

Projection	PSNR (dB)	Size (MB)
None (raw hash features)	27.62	15.21
Linear	27.76	14.28
MLP (ours)	27.84	13.61

Context dimension D^c from 12 to 48 changes size by at most 0.35 MB (Mip-NeRF360).

Learned compact context: +0.22 dB and 1.6 MB smaller than raw hash features.

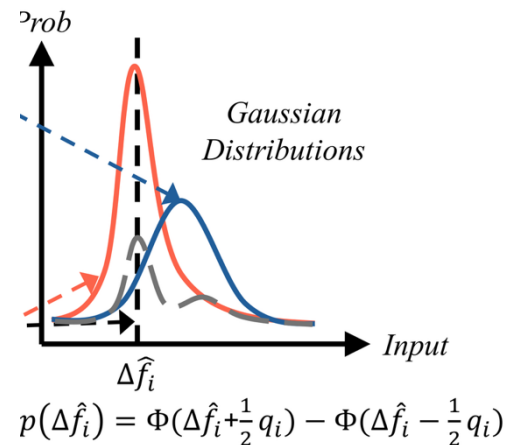
(2) Adaptive Residual Quantization



$$\Delta \mathbf{f}_i = \mathbf{f}_i - \mu_i^s, \quad \mathbf{r}_i = \lfloor \Delta \mathbf{f}_i \oslash \mathbf{q}_i \rfloor \in \mathbb{Z}^{D^f},$$

$$\mathbf{q}_i = q_0 (1 + \tanh \Delta q_i), \quad \hat{\mathbf{f}}_i = \mu_i^s + \mathbf{r}_i \odot \mathbf{q}_i$$

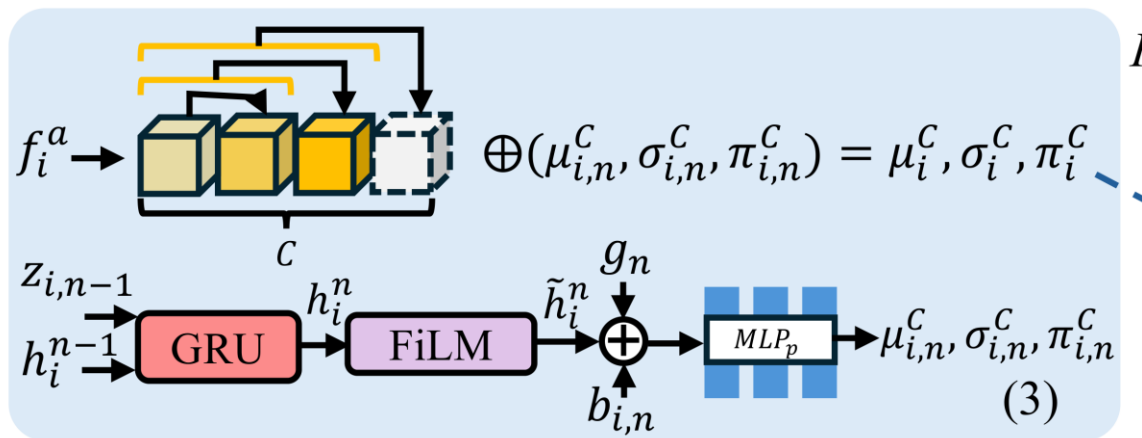
- Predict the mean μ^s and a quantization step adjustment from the compact context, then encode only the integer residual r .
- Centering attributes around the predicted mean narrows their range and reduces symbol entropy.
- During training, use uniform noise of width q to approximate quantization and allow gradients to flow.



Anchor-feature symbols	Attribute	Residual
Symbol range	1,023	287
Empirical entropy (bits)	8.27	5.22
Coded bits / element	8.44	5.38

Residual coding cuts coded bits per element from 8.44 to 5.38 (−36%).

(3) FiLM-Modulated Autoregressive GRU



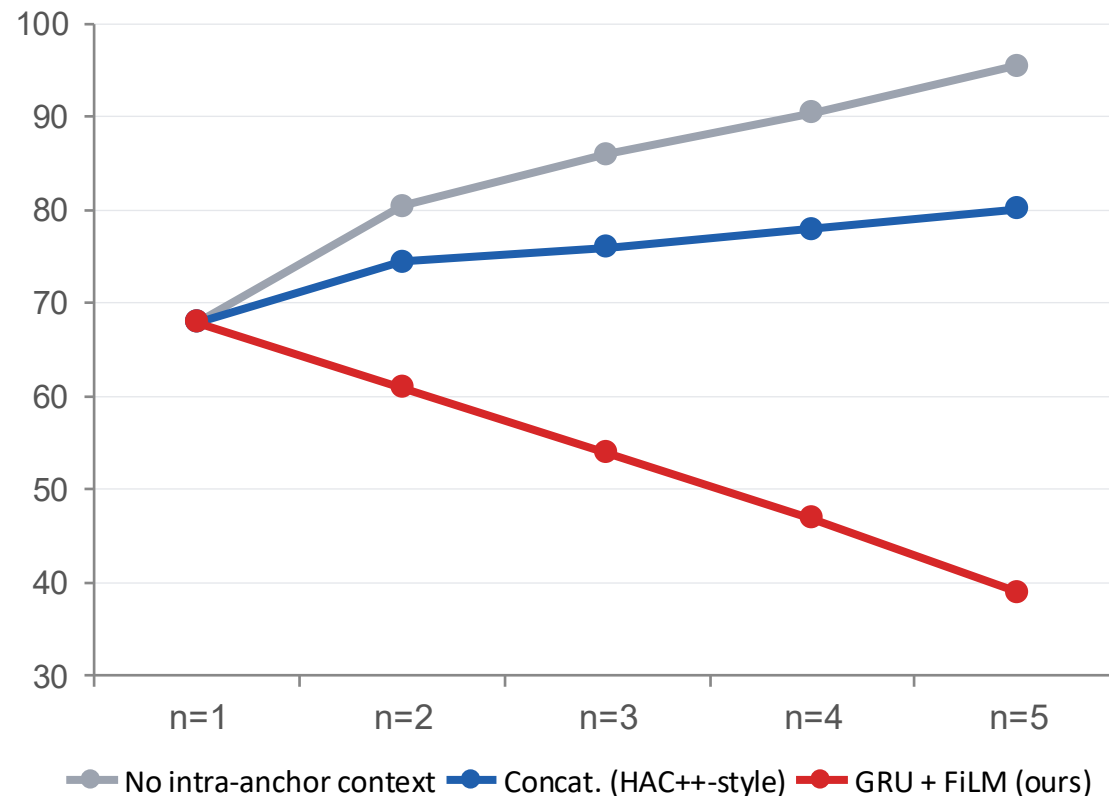
$$h_i^n = \text{GRU}(z_{i,n-1}, h_i^{n-1}), \quad h_i^1 = \text{MLP}_{\text{cxt}}(s_i)$$

$$\tilde{h}_i^n = h_i^n \odot (\mathbf{1} + \tanh \gamma_i^n) + \beta_i^n$$

$$\mu_{i,n}^C, \sigma_{i,n}^C, \pi_{i,n}^C = \text{MLP}_p(\tilde{h}_i^n \oplus b_{i,n} \oplus g_n)$$

- Split each anchor feature into chunks and update the GRU after decoding each chunk, keeping the input cost constant per step.
- FiLM adapts the hidden state to each anchor's base statistics. A shared predictor uses a position embedding to distinguish chunks.

Anchor-feature bits per anchor, by chunk (Mip-NeRF360)



Concatenation gets costlier with each chunk, and the GRU gets cheaper (269 vs 377 bits).

Residual Mixture Entropy Model and Training

Mixture of the base (spatial) and refined (causal) components:

$$p(\Delta \hat{\mathbf{f}}_i^a) = \prod_{j=1}^{D^f} \sum_{k \in \{s, c\}} \pi_{i,j}^k \left[\Phi\left(\Delta \hat{\mathbf{f}}_{i,j}^a + \frac{q_{i,j}}{2}; \mu_{i,j}^k, \sigma_{i,j}^k\right) - \Phi\left(\Delta \hat{\mathbf{f}}_{i,j}^a - \frac{q_{i,j}}{2}; \mu_{i,j}^k, \sigma_{i,j}^k\right) \right]$$

- The base component (μ^s, σ^s) predicts coarse statistics from the compact context.
- The refined component adjusts probabilities based on previously decoded chunks within the anchor.
- Arithmetic coding uses approximately $-\log_2 p$ bits per symbol.

Training objective (rendering + rate):

$$\mathcal{L}_{\text{total}} = (1 - \lambda_{\text{SSIM}}) \mathcal{L}_1 + \lambda_{\text{SSIM}} \mathcal{L}_{\text{SSIM}} + \lambda_{\text{vol}} \mathcal{L}_{\text{vol}} + \frac{\lambda_r}{N N_p} (\mathcal{L}_{\Delta \text{entropy}} + \mathcal{L}_{\text{hash}})$$

λ_r sweeps the rate–distortion trade-off, and learnable masks prune inactive anchors and offsets.

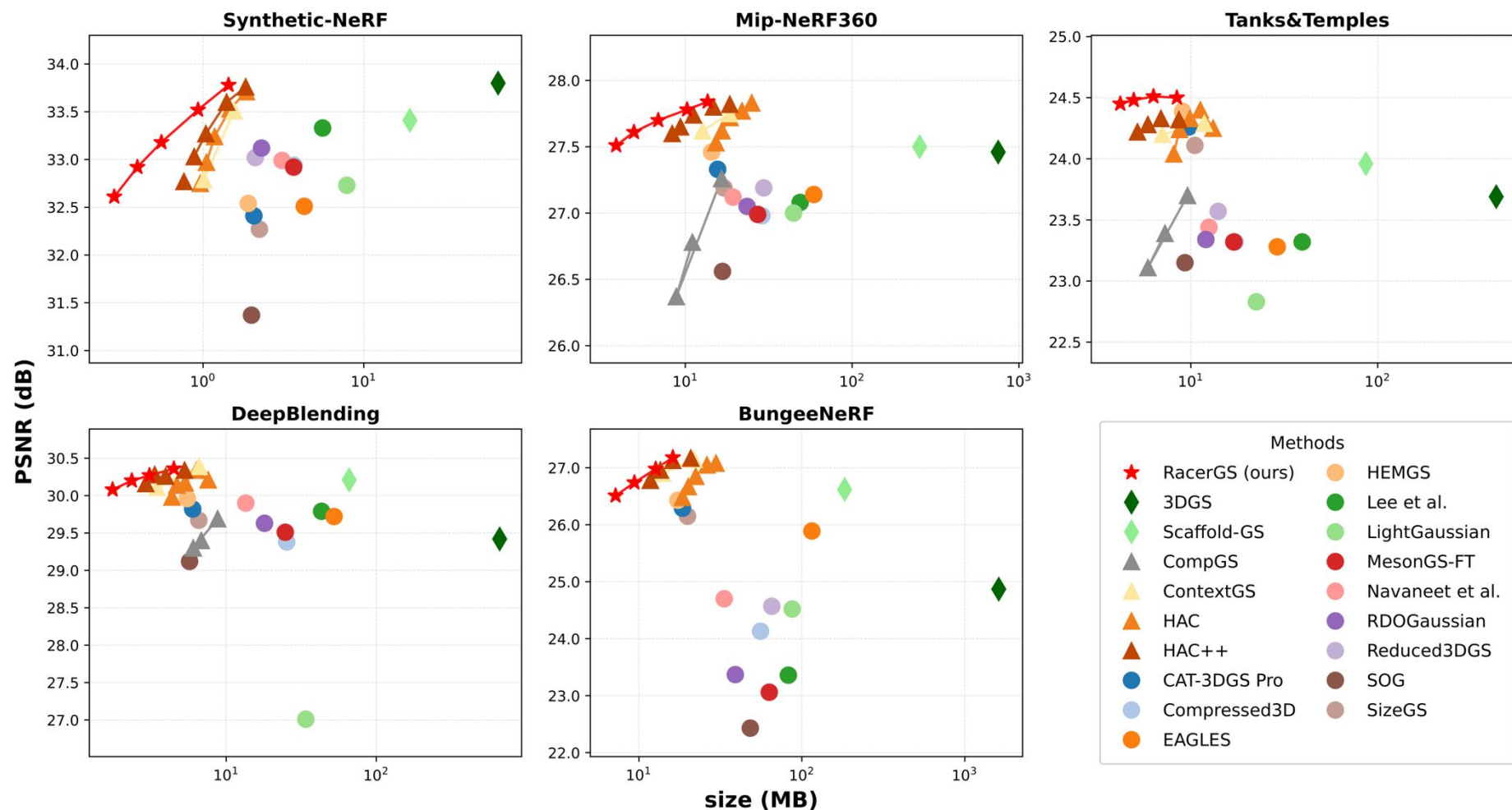
Main Results

Method	Mip-NeRF360		Tanks&Temples		DeepBlending		BungeeNeRF	
	PSNR	MB	PSNR	MB	PSNR	MB	PSNR	MB
3DGS	27.46	750.9	23.69	431.0	29.42	663.9	24.87	1616
Scaffold-GS	27.50	253.9	23.96	86.50	30.21	66.00	26.62	183.0
HAC (hr)	27.77	21.87	24.40	11.24	30.34	6.35	27.08	29.72
ContextGS (hr)	27.75	18.41	24.29	11.80	30.39	6.60	27.15	21.80
HAC++ (lr)	27.60	8.34	24.22	5.18	30.16	2.91	26.78	11.75
HAC++ (hr)	27.82	18.48	24.32	8.63	30.34	5.28	27.17	20.82
RacerGS (lr)	27.51	3.85	24.45	4.19	30.08	1.75	26.51	7.19
RacerGS (hr)	27.84	13.61	24.50	8.42	30.36	4.48	27.18	16.15

Subset of Table 1, the paper compares 18 baselines on 5 datasets.

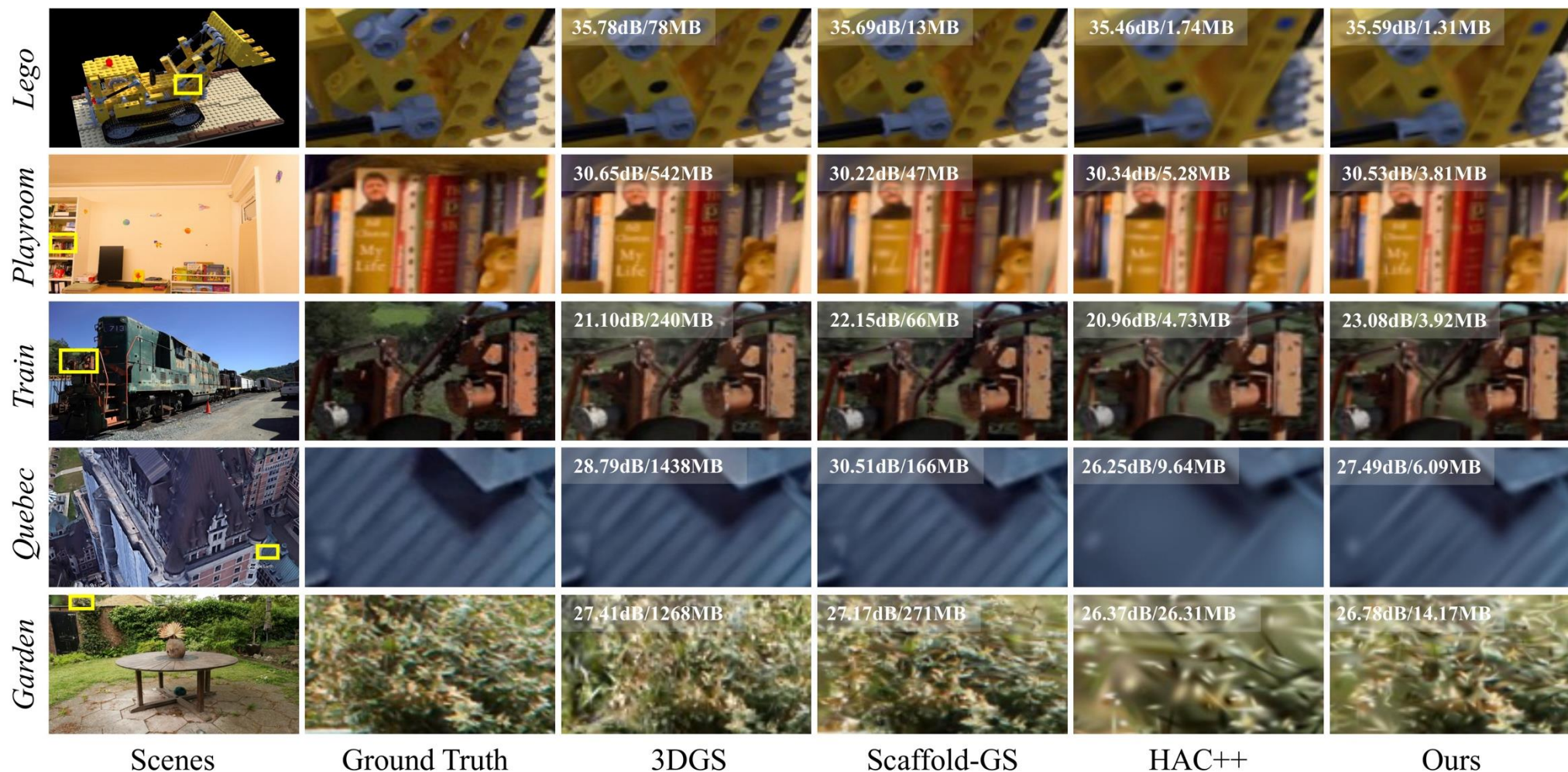
Smallest size on every dataset, and 26% smaller than HAC++ at equal PSNR on Mip-NeRF360.

Rate–Distortion Curves



RacerGS (red stars) gives the best rate–distortion trade-off on all five datasets.

Qualitative Comparison



Fewer artifacts and sharper details than 3DGS, Scaffold-GS and HAC++ at equal or smaller size.

Ablation Study (Mip-NeRF360)

Removing each component

Variant	PSNR	Size (MB)	FPS
w/o residual quantization	27.83	18.24	146
w/o hash projection	27.62	15.21	107
Concat. instead of GRU	27.81	16.87	156
GRU w/o FiLM	27.75	17.32	149
RacerGS (full)	27.84	13.61	171

Controlled intra-anchor context comparison

Context model	Params	PSNR	MB
No intra-anchor context	0	27.69	18.20
Concat. (HAC++-style)	≈40k	27.81	16.87
Concat. (capacity-matched)	≈48k	27.82	15.24
GRU w/o FiLM	≈48k	27.75	17.32
GRU + FiLM (ours)	≈48k	27.84	13.61

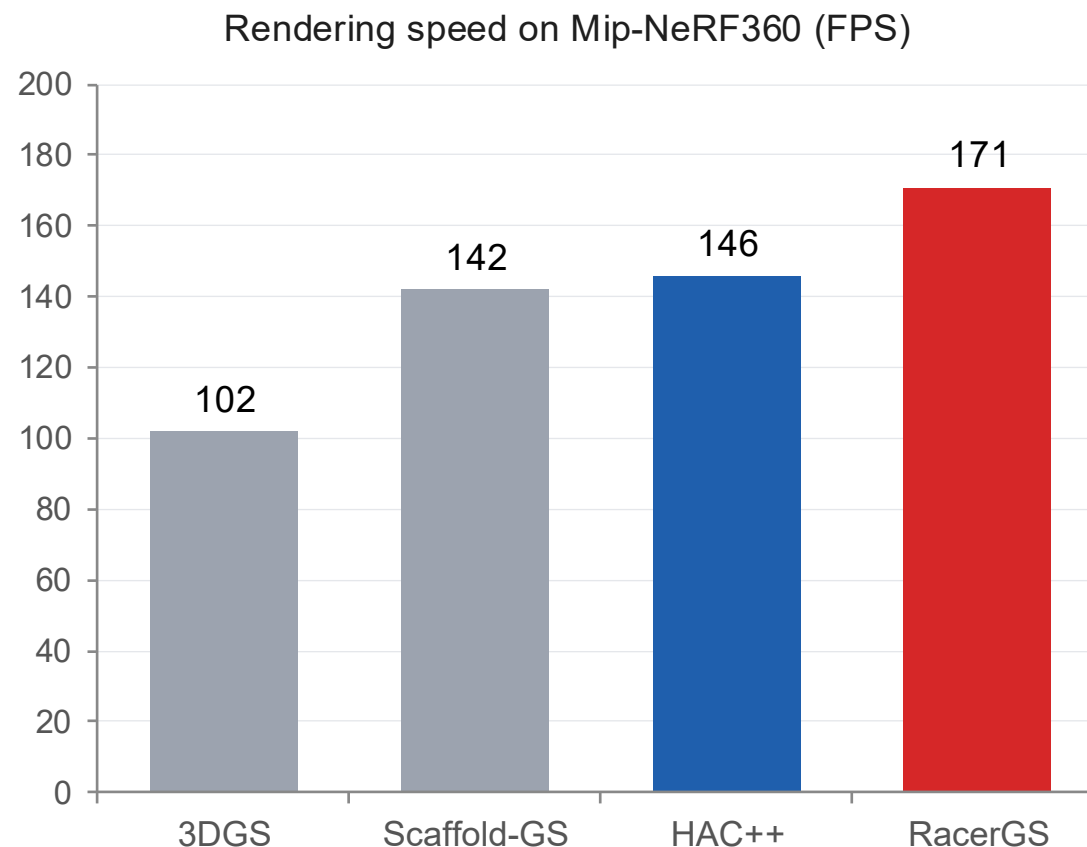
- Every component helps, with residual quantization reducing size the most (18.24 → 13.61 MB).
- At the same parameter budget, GRU + FiLM saves 1.6 MB over concatenation, showing the benefit of combining recurrence with FiLM.

The full model is the smallest, the most accurate and the fastest to render.

Efficiency

Mip-NeRF360	Train (s)	Peak mem. (GB)	FPS
3DGS	1607	—	102
Scaffold-GS	1350	15.8	142
HAC++	2447	17.4	146
RacerGS	2850	13.9	171

Dataset	Encode (s)	Decode (s)
Synthetic-NeRF	0.63–1.28	0.89–1.75
Mip-NeRF360	4.24–8.59	7.40–14.20
Tanks&Temples	4.53–7.48	7.80–12.63
DeepBlending	1.85–3.01	4.03–6.25
BungeeNeRF	10.03–16.26	16.43–25.98



Fastest rendering and lowest memory, and training is slower than HAC++ (2850 vs 2447 s).

Conclusion

- **RacerGS** compresses 3DGS by making better predictions for each anchor before encoding it.
- **Three components:** compact hash projection, adaptive residual quantization, and an autoregressive GRU with FiLM conditioning.
- **Results:** over $150\times$ smaller than 3DGS and $28\times$ smaller than Scaffold-GS at comparable quality, with better rate–distortion performance than prior 3DGS codecs across five datasets.
- **Limitation:** training takes longer than HAC++. Future work can explore context models that support parallel processing and rate control that adapts to scene content.



Project page

racergs.github.io

Thank you!